Name : Avni Sameer Dongre

Roll no : A4_B3_46

DAA Practical 5 :

Aim: Implement a dynamic algorithm for Longest Common Subsequence (LCS) to find the

length and LCS for DNA sequences.

Problem Statement:

(i) DNA sequences can be viewed as strings of A, C, G, and T characters, which represent nucleotides. Finding the similarities between two DNA sequences are an important computation performed in bioinformatics.

[Note that a subsequence might not include consecutive elements of the original sequence.]

TASK 1: Find the similarity between the given X and Y sequence.

X=AGCCCTAAGGGCTACCTAGCTT

Y= GACAGCCTACAAGCGTTAGCTTG

Code:

```
#include <stdio.h>
#include <string.h>

#define MAX 100

void LCS_Length(char X[], char Y[], int m, int n, int c[MAX][MAX], char b[MAX][MAX]) {
    for (int i = 0; i <= m; i++)
        c[i][0] = 0;
    for (int j = 0; j <= n; j++)
        c[0][j] = 0;
```

```
    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            if (X[i - 1] == Y[j - 1]) {
                c[i][j] = c[i - 1][j - 1] + 1;
                b[i][j] = '\\';
            } else if (c[i - 1][j] >= c[i][j - 1]) {
                c[i][j] = c[i - 1][j];
                b[i][j] = '^';
            } else {
                c[i][j] = c[i][j - 1];
                b[i][j] = '<';
            }
        }
    }
}

void Print_LCS(char b[MAX][MAX], char X[], int i, int j) {
    if (i == 0 || j == 0)
        return;
    if (b[i][j] == '\\') {
        Print_LCS(b, X, i - 1, j - 1);
        printf("%c", X[i - 1]);
    } else if (b[i][j] == '^') {
        Print_LCS(b, X, i - 1, j);
    } else {
        Print_LCS(b, X, i, j - 1);
    }
}

int main() {
```

```
char X[MAX], Y[MAX];
int c[MAX][MAX];
char b[MAX][MAX];

printf("Enter first DNA sequence: ");
scanf("%s", X);
printf("Enter second DNA sequence: ");
scanf("%s", Y);

int m = strlen(X);
int n = strlen(Y);

LCS_Length(X, Y, m, n, c, b);

printf("\nCost Matrix (c):\n");
for (int i = 0; i <= m; i++) {
   for (int j = 0; j <= n; j++) {
      printf("%2d ", c[i][j]);
   }
   printf("\n");
}

printf("\nDirection Matrix (b):\n");
for (int i = 1; i <= m; i++) {
   for (int j = 1; j <= n; j++) {
      printf(" %c ", b[i][j]);
   }
   printf("\n");
}

printf("\nLongest Common Subsequence: ");
```

```
    Print_LCS(b, X, m, n);
    printf("\nLength of LCS: %d\n", c[m][n]);

    return 0;
}
```

Output:

Output

Clear

```
Enter first DNA sequence: GCCCTAAGGGCTACCTAGCTT
Enter second DNA sequence: GACAGCCTACAAGCGTTAGCTTG

Cost Matrix (c):
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
 0  1  1  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
 0  1  1  2  2  2  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
 0  1  1  2  2  2  3  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
 0  1  1  2  2  2  3  4  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
 0  1  2  2  3  3  3  4  5  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
 0  1  2  2  3  3  3  4  5  6  6  7  7  7  7  7  7  7  7  7  7  7  7  7
 0  1  2  2  3  4  4  4  5  6  6  7  7  8  8  8  8  8  8  8  8  8  8  8
 0  1  2  2  3  4  4  4  5  6  6  7  7  8  8  9  9  9  9  9  9  9  9  9
 0  1  2  2  3  4  4  4  5  6  6  7  7  8  8  9  9  9  9 10 10 10 10 10
 0  1  2  3  3  4  5  5  5  6  7  7  7  8  9  9  9  9  9 10 11 11 11 11
 0  1  2  3  3  4  5  5  6  6  7  7  7  8  9  9 10 10 10 10 11 12 12 12
 0  1  2  3  4  4  5  5  6  7  7  8  8  8  9  9 10 10 11 11 11 12 12 12
 0  1  2  3  4  4  5  6  6  7  8  8  8  8  9  9 10 10 11 11 12 12 12 12
 0  1  2  3  4  4  5  6  6  7  8  8  8  8  9  9 10 10 11 11 12 12 12 12
 0  1  2  3  4  4  5  6  7  7  8  8  8  8  9  9 10 11 11 11 12 13 13 13
 0  1  2  3  4  4  5  6  7  8  8  9  9  9  9  9 10 11 12 12 12 13 13 13
 0  1  2  3  4  5  5  6  7  8  8  9  9 10 10 10 10 11 12 13 13 13 13 14
 0  1  2  3  4  5  6  6  7  8  9  9  9 10 11 11 11 11 12 13 14 14 14 14
 0  1  2  3  4  5  6  6  7  8  9  9  9 10 11 11 12 12 12 13 14 15 15 15
 0  1  2  3  4  5  6  6  7  8  9  9  9 10 11 11 12 13 13 13 14 15 16 16
```

```
Direction Matrix (b):
 \  <  <  <  \  <  <  <  <  <  <  <  \  <  \  <  <  <  \  <  <  <  \
 ^  ^  \  <  <  \  \  <  <  \  <  <  <  \  <  <  <  <  <  \  <  <  <
 ^  ^  \  ^  ^  \  \  <  <  \  <  <  <  \  <  <  <  <  <  \  <  <  <
 ^  ^  \  ^  ^  \  \  <  <  \  <  <  <  \  <  <  <  <  <  \  <  <  <
 ^  ^  ^  ^  ^  ^  ^  \  <  <  <  <  <  <  <  \  \  <  <  <  \  \  <
 ^  \  ^  \  <  ^  ^  ^  \  <  \  \  <  <  <  <  <  \  <  <  <  <  <
 ^  \  ^  \  ^  ^  ^  ^  \  ^  \  \  <  <  <  <  <  \  <  <  <  <  <
 \  ^  ^  ^  \  <  ^  ^  ^  ^  ^  ^  \  <  \  <  <  <  \  <  <  <  \
 \  ^  ^  ^  \  ^  ^  ^  ^  ^  ^  ^  \  ^  \  <  <  <  \  <  <  <  \
 \  ^  ^  ^  \  ^  ^  ^  ^  ^  ^  ^  \  ^  \  ^  ^  ^  \  <  <  <  \
 ^  ^  \  ^  ^  \  \  ^  ^  \  ^  ^  ^  \  ^  ^  ^  ^  ^  \  <  <  <
 ^  ^  ^  ^  ^  ^  ^  \  ^  ^  ^  ^  ^  ^  ^  \  \  <  ^  ^  \  \  <
 ^  \  ^  \  ^  ^  ^  ^  \  ^  \  \  ^  ^  ^  ^  ^  \  <  ^  ^  ^  ^
 ^  ^  \  ^  ^  \  \  ^  ^  \  ^  ^  ^  \  ^  ^  ^  ^  ^  \  ^  ^  ^
 ^  ^  \  ^  ^  \  \  ^  ^  \  ^  ^  ^  \  ^  ^  ^  ^  ^  \  ^  ^  ^
 ^  ^  ^  ^  ^  ^  ^  \  ^  ^  ^  ^  ^  ^  ^  \  \  ^  ^  ^  \  \  <
 ^  \  ^  \  ^  ^  ^  ^  \  ^  \  \  <  ^  ^  ^  ^  \  <  ^  ^  ^  ^
 \  ^  ^  ^  \  ^  ^  ^  ^  ^  ^  ^  \  <  \  ^  ^  ^  \  <  ^  ^  \
 ^  ^  \  ^  ^  \  \  ^  ^  \  ^  ^  ^  \  <  <  ^  ^  ^  \  <  <  ^
 ^  ^  ^  ^  ^  ^  ^  \  ^  ^  ^  ^  ^  ^  ^  \  \  ^  ^  ^  \  \  <
 ^  ^  ^  ^  ^  ^  ^  \  ^  ^  ^  ^  ^  ^  ^  \  \  <  ^  ^  \  \  <

Longest Common Subsequence: GCCCTAAGGTTAGCTT
Length of LCS: 16
```

TASK-2: Find the longest repeating subsequence (LRS). Consider it as a variation of the longest common subsequence (LCS) problem.

Let the given string be S. You need to find the LRS within S. To use the LCS framework, you effectively compare S with itself. So, consider string1 = S and string2 = S.

Example:

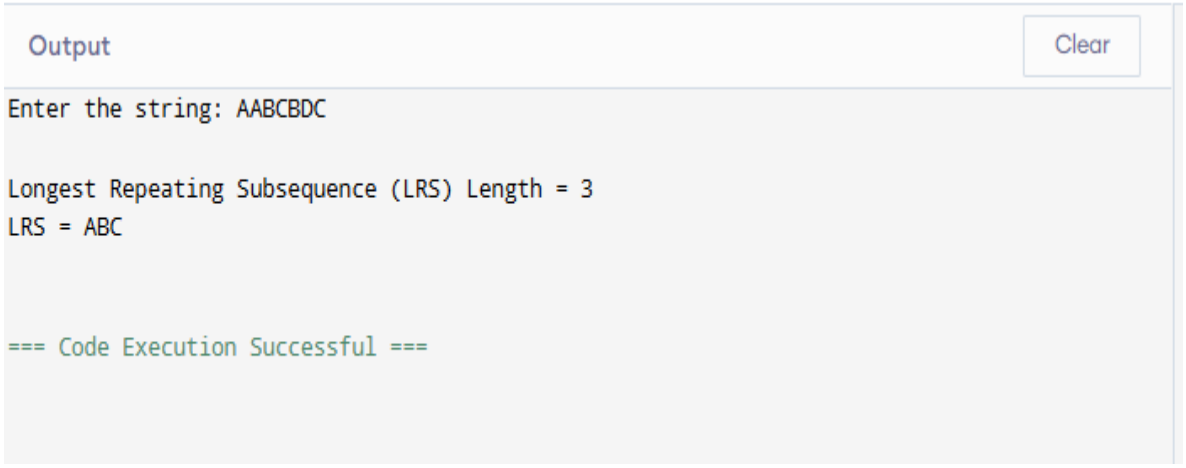
AABCBDC

Code:

```
#include <stdio.h>
#include <string.h>
#define MAX 100

int main() {
    char S[MAX];
    printf("Enter the string: ");
    scanf("%s", S);
```

```
int n = strlen(S);
int dp[MAX+1][MAX+1];

// Initialize
for (int i = 0; i <= n; i++)
    for (int j = 0; j <= n; j++)
        dp[i][j] = 0;

// Build DP table
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
        if (S[i-1] == S[j-1] && i != j)
            dp[i][j] = 1 + dp[i-1][j-1];
        else
            dp[i][j] = dp[i-1][j] > dp[i][j-1] ? dp[i-1][j] : dp[i][j-1];
    }
}

// Reconstruct LRS
int i = n, j = n;
char lrs[MAX];
int k = 0;
while (i > 0 && j > 0) {
    if (S[i-1] == S[j-1] && i != j) {
        lrs[k++] = S[i-1];
        i--; j--;
    }
    else if (dp[i-1][j] > dp[i][j-1])
        i--;
    else
        j--;
}
lrs[k] = '\0';

// Reverse the string
for (int a = 0; a < k/2; a++) {
    char temp = lrs[a];
    lrs[a] = lrs[k-1-a];
    lrs[k-1-a] = temp;
}

printf("\nLongest Repeating Subsequence (LRS) Length = %d\n", dp[n][n]);
printf("LRS = %s\n", lrs);

return 0;
```

Output:



Leetcode Assessment :

Problem Statement:

1143. Longest Common Subsequence

Medium | Topics | Companies | Hint

Given two strings `text1` and `text2`, return *the length of their longest* ***common subsequence***. If there is no **common subsequence**, return `0`.

A **subsequence** of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters.

- For example, `"ace"` is a subsequence of `"abcde"`.

A **common subsequence** of two strings is a subsequence that is common to both strings.

Example 1:

```
Input: text1 = "abcde", text2 = "ace"
Output: 3
Explanation: The longest common subsequence is "ace" and its length is 3.
```

Example 2:

```
Input: text1 = "abc", text2 = "abc"
Output: 3
Explanation: The longest common subsequence is "abc" and its length is 3.
```

Example 3:

```
Input: text1 = "abc", text2 = "def"
Output: 0
```

Explanation: The longest common subsequence is "abc" and its length is 3.

Example 3:

```
Input: text1 = "abc", text2 = "def"
Output: 0
```
Explanation: There is no such common subsequence, so the result is 0.

Constraints:

- `1 <= text1.length, text2.length <= 1000`
- `text1` and `text2` consist of only lowercase English characters.

Code:

Code

C Auto

```c
int longestCommonSubsequence(char * text1, char * text2) {
    int m = strlen(text1), n = strlen(text2);
    int **dp = (int **)malloc((m + 1) * sizeof(int *));
    for (int i = 0; i <= m; i++) {
        dp[i] = (int *)malloc((n + 1) * sizeof(int));
        for (int j = 0; j <= n; j++) dp[i][j] = 0;
    }
    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            if (text1[i - 1] == text2[j - 1])
                dp[i][j] = dp[i - 1][j - 1] + 1;
            else
                dp[i][j] = dp[i - 1][j] > dp[i][j - 1] ? dp[i - 1][j] : dp[i][j - 1];
        }
    }
    int result = dp[m][n];
    for (int i = 0; i <= m; i++) free(dp[i]);
    free(dp);
    return result;
}

```

Output: